MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2016/2017

EEN7146 ADVANCED MICROPROCESSOR SYSTEM

8 OCT 2016
9.00 A.M. – 12.00 P.M.
(3 Hours)

INSTRUCTIONS TO STUDENTS

1. This Question paper consists of 12 pages with 4 Questions only.
2. Attempt **ALL** questions. All questions carry equal marks and the distribution of the marks for each question is given.
3. Please print all your answers in the Answer Booklet provided.
4. The summary of instruction set and the formats of some relevant descriptors and table entry are provided in Appendices A, B, C, D and E, respectively.

Question 1

(a) List FOUR major characteristics of a Complex Instruction Set Computer (CISC) and a Reduced Instruction Set Computer (RISC) microprocessor system.

[4 marks]

(b) Explain what System on Chip (SoC) is. Contrast the difference between SoC and System in a Package (SiP).

[3 marks]

(c) Explain how Thumb instruction set can reduce the code density for Advance RISC Machine (ARM) microprocessors and discuss its limitations compared to the ARM instruction set.

[4 marks]

(d) The 8086 microprocessor is Intel's first 16-bit microprocessor

(i) Determine the maximum size of the addressable memory space in an 8086 based computer. How much memory can be active at a time in this computer? How to activate other parts of the memory?

[3 marks]

(ii) A 16-bit integer number is stored in the external memory of an 8086 based computer, as shown in Figure Q1(d). Determine its equivalent decimal value if this number is represented using the two's complement notation.

[2 marks]

External memory

Address	Data
2020AH	32H
2020BH	89H

Figure Q1(d)

(e) Protection is a concept where memory is safeguarded against unauthorized access.

(i) Contrast the code and data protection rule.

(ii) Briefly explain the function of a call gate.

[4 marks]

(f) What is multithreading? Explain how multithreading can be carried out in time-slicing method and multi-core method.

[1 mark]

[4 marks]

Continued ...

Question 2

(a) The current values of SS and SP in an Intel 8086 microprocessor are 2C00H and FF40H, respectively.

(i) Calculate the physical addresses of the current bottom of stack and top of stack. [4 marks]

(ii) Determine the number of data words that could possibly be held in the stack based on the current SS:SP. [2 marks]

(iii) Given that the CX register contains A41CH, identify the affected memory locations and their new contents if the CX register is pushed on to the stack. [1 mark]

(b) If the contents of registers in an 8086 microprocessor are given in Table Q2(b), determine the addressing mode, physical address, and the content of the DESTINATION operand in each instruction below:

(i) `MOV [3030H], 0BH`

(ii) `MOV ES:[BX][DI]+0110H, CX` [2 marks]

[3 marks]

Table Q2(b)

Register	Contents
AX	1100H
BX	5446H
CX	9800H
DI	5A5AH

Register	Contents
CS	1000H
DS	2100H
ES	3200H
SS	4000H

(c) Find the final values of AX and BX after executing the following instruction sequences.

```
MOV     AX, 32AAH
MOV     BX, 1234H
CBW
XOR     BX, 0FAFAH
STC
SBB     AX, BX
```

[3 marks]

(d) Using 8086 assembly instruction set, write a program to calculate the number of '1' in AL register. Store the result in BL register. *[For example, if the content of AL register is 01111001B, number of '1' is 5].*

[4 marks]

(e) Develop a real-mode IA-32 instruction sequence to convert all lower case characters in a string stored in physical address starting from 20100H to upper case and store the results in physical address starting from 20400H. Assume that the string is terminated with a null character and the total length include termination will not exceed 700 characters. *[For example, "een7146 - Advanced Microprocessor System" is to be converted to "EEN7146 - ADVANCED MICROPROCESSOR SYSTEM"].*

[6 marks]

Continued ...

Question 3

(a) A Global Descriptor Table (GDT) is to be initialized with a base address of 12003300H. The table must be able to store 800 descriptors.

(i) Determine the value to be loaded into the Global Descriptor Table Register (GDTR).

[2 marks]

(ii) If a value of 1020H is loaded into the Local Descriptor Table Register (LDTR), determine the address range of the Local Descriptor Table (LDT) descriptor.

[1 mark]

(b) Given that the GDT base address is 80109260H, and the contents of the registers and memory are as shown below in Table Q3(b). If the instruction `MOV [ESI],EDX` is executed, determine:

(i) The linear address of the instruction.

[5 marks]

(ii) The linear address of the destination operand.

[8 marks]

(iii) The size of the data segment in part (b)(ii).

[2 marks]

(iv) The privilege levels allowed to access the data segment of part (b)(ii). Justify your answer.

[3 marks]

Table Q3(b)

Register	Contents
LDTR	0030H
EIP	0000 0A80H
EDX	0000 1000H
ESI	0000 3000H

Register	Contents
CS	0018H
DS	0027H
ES	0153H
SS	0253H

Linear address	Contents (hex)			
	+0	+2	+4	+6
80109250H	0000	2000	FB10	A01F
80109258H	0000	A000	FB10	B01F
80109260H	0000	A000	FB10	A01F
80109268H	0100	AC58	8240	0000
80109270H	0100	AC00	8240	2000
80109278H	32FF	5000	FB40	3200
80109280H	0000	C000	FB10	A01F
80109288H	0000	0000	0000	0000
80109290H	01FF	7800	8298	4100
80109298H	B000	20C0	FB10	8A1F

Linear address	Contents (hex)			
	+0	+2	+4	+6
41987800H	0000	0000	0000	0000
41987808H	0000	0000	0000	0000
41987810H	0000	0000	0000	0000
41987818H	2000	2000	FB10	801F
41987820H	32FF	0000	F640	A000
41987828H	0000	1000	FB10	801F
41987830H	0000	0000	0000	0000
41987838H	0000	2000	F110	A01F
41987840H	0000	0000	0000	0000
41987848H	0000	0000	0000	0000

(c) Explain the sequence of switching from real mode to protected mode.

[4 marks]

Continued ...

Question 4

(a) Given that the contents of the MMX registers are:

$[MM_1]$ = 90A0 00A0 10F5 0032 H
$[MM_2]$ = 00B5 0090 01A2 A824 H
$[MM_3]$ = A0B1 2311 1400 2310 H

(i) Evaluate the contents of MM_1, MM_2, and MM_3 after the following sequence of instructions are executed:

```
PACKUSWB    MM1, MM2
PADDUSB     MM3, MM1
```

Note: Show all necessary working procedures.

[6 marks]

(ii) Using the content of MM_1 register, determine the values of the individual elements for 8×8, 4×16, and 2×32 data.

[3 marks]

(b) The Streaming SIMD Extensions (SSE) instructions are divided into four functional groups.

(i) List these FOUR functional groups.

[2 marks]

(ii) Using a block diagram, describe the operations of the following SSE instructions:

```
ADDPS xmm1, xmm2/m128
SUBSS xmm1, xmm2/m32
```

[6 marks]

(iii) Using a suitable example, discuss how SSE instructions can improve the performance of floating point calculation compared to conventional instructions.

[4 marks]

(c) The Intel Core microarchitecture is the foundation for Intel architecture-based desktop, mobile, and mainstream server multi-core processors. Discuss the concept of Intel Advanced Smart Cache.

[4 marks]

Continued ...

Appendix A

Instruction Set Summary for Intel 8086/80186/80286/80386/80486

AAA - Ascii Adjust for Addition
AAS - Ascii Adjust for Subtraction
ADC - Add With Carry
ADD - Arithmetic Addition
AND - Logical And
BSF - Bit Scan Forward (386+)
BSR - Bit Scan Reverse (386+)
BSWAP - Byte Swap (486+)
BT - Bit Test (386+)
BTC - Bit Test with Compliment (386+)
BTR - Bit Test with Reset (386+)
BTS - Bit Test and Set (386+)
CALL - Procedure Call
CBW - Convert Byte to Word
CDQ - Convert Double to Quad (386+)
CLC - Clear Carry
CLD - Clear Direction Flag
CLI - Clear Interrupt Flag (disable)
CLTS - Clear Task Switched Flag (286+ privileged)
CMC - Complement Carry Flag
CMP - Compare
CMPS - Compare String (Byte, Word or Doubleword)
CMPXCHG - Compare and Exchange
CWD - Convert Word to Doubleword
CWDE - Convert Word to Extended Doubleword (386+)
DAA - Decimal Adjust for Addition
DAS - Decimal Adjust for Subtraction
DEC - Decrement
DIV - Divide
ENTER - Make Stack Frame (80188+)
ESC - Escape
HLT - Halt CPU
IDIV - Signed Integer Division
IMUL - Signed Multiply
IN - Input Byte or Word From Port
INC - Increment
INT - Interrupt
INTO - Interrupt on Overflow
INVD - Invalidate Cache (486+)
INVLPG - Invalidate Translation Look-Aside Buffer Entry (486+)
IRET/IRETD - Interrupt Return
Jxx - Jump Instructions Table
JCXZ/JECXZ - Jump if Register (E)CX is Zero
JMP - Unconditional Jump
LAHF - Load Register AH From Flags
LAR - Load Access Rights (286+ protected)
LDS - Load Pointer Using DS
LEA - Load Effective Address
LES - Load Pointer Using ES
LFS - Load Pointer Using FS (386+)
LGDT - Load Global Descriptor Table (286+ privileged)
LIDT - Load Interrupt Descriptor Table (286+ privileged)
LGS - Load Pointer Using GS (386+)
LLDT - Load Local Descriptor Table (286+ privileged)
LMSW - Load Machine Status Word (286+ privileged)

Continued ...

LOCK - Lock Bus
LODS - Load String (Byte, Word or Double)
LOOP - Decrement CX and Loop if CX Not Zero
LOOPE/LOOPZ - Loop While Equal / Loop While Zero
LOOPNZ/LOOPNE - Loop While Not Zero / Loop While Not Equal
LSL - Load Segment Limit (286+ protected)
LSS - Load Pointer Using SS (386+)
LTR - Load Task Register (286+ privileged)
MOV - Move Byte or Word
MOVS - Move String (Byte or Word)
MOVSX - Move with Sign Extend (386+)
MOVZX - Move with Zero Extend (386+)
MUL - Unsigned Multiply
NEG - Two's Complement Negation
NOP - No Operation (90h)
OR - Inclusive Logical OR
OUT - Output Data to Port
POP - Pop Word off Stack
POPA/POPAD - Pop All Registers onto Stack (80188+)
POPF/POPFD - Pop Flags off Stack
PUSH - Push Word onto Stack
PUSHA/PUSHAD - Push All Registers onto Stack (80188+)
PUSHF/PUSHFD - Push Flags onto Stack
RCL - Rotate Through Carry Left
RCR - Rotate Through Carry Right
REP - Repeat String Operation
REPE/REPZ - Repeat Equal / Repeat Zero
REPNE/REPNZ - Repeat Not Equal / Repeat Not Zero
RET/RETF - Return From Procedure
ROL - Rotate Left
ROR - Rotate Right
SAHF - Store AH Register into FLAGS
SAL/SHL - Shift Arithmetic Left / Shift Logical Left
SAR - Shift Arithmetic Right
SBB - Subtract with Borrow/Carry
SCAS - Scan String (Byte, Word or Doubleword)
SETS - Set if Signed (386+)
SETNS - Set if Not Signed (386+)
SETC - Set if Carry (386+)
SETNC - Set if Not Carry (386+)
SETO - Set if Overflow (386+)
SETP/SETPE - Set if Parity / Set if Parity Even (386+)
SETNP/SETPO - Set if No Parity / Set if Parity Odd (386+)
SGDT - Store Global Descriptor Table (286+ privileged)
SIDT - Store Interrupt Descriptor Table (286+ privileged)
SHL - Shift Logical Left
SHR - Shift Logical Right
SHLD/SHRD - Double Precision Shift (386+)
SLDT - Store Local Descriptor Table (286+ privileged)
SMSW - Store Machine Status Word (286+ privileged)
STC - Set Carry
STD - Set Direction Flag
STI - Set Interrupt Flag (Enable Interrupts)
STOS - Store String (Byte, Word or Doubleword)
STR - Store Task Register (286+ privileged)
SUB - Subtract
TEST - Test for Bit Pattern
WBINVD - Write-Back and Invalidate Cache (486+)
XCHG - Exchange
XOR - Exclusive OR

Continued ...

Mnemonic	Meaning	Format	Operation
LOOP	Loop	LOOP Short-label	(CX) ← (CX) − 1 Jump is initiated to location defined by short-label if (CX) ≠ 0; otherwise, execute next sequential instruction
LOOPE/LOOPZ	Loop while equal/ loop while zero	LOOPE/LOOPZ Short-label	(CX) ← (CX) − 1 Jump to location defined by short-label if (CX) ≠ 0 and (ZF) = 1; otherwise, execute next sequential instruction
LOOPNE/ LOOPNZ	Loop while not equal/ loop while not zero	LOOPNE/LOOPNZ Short-label	(CX) ← (CX) − 1 Jump to location defined by short-label if (CX) ≠ 0 and (ZF) = 0; otherwise, execute next sequential instruction

Mnemonic	Meaning	Format	Operation	Flags Affected
MOVS	Move string	MOVSB/MOVSW	((ES)0 + (DI)) ← ((DS)0 + (SI)) (SI) ← (SI) ± 1 or 2 (DI) ← (DI) ± 1 or 2	None
CMPS	Compare string	CMPSB/CMPSW	Set flags as per ((DS)0 + (SI)) − ((ES)0 + (DI)) (SI) ← (SI) ± 1 or 2 (DI) ← (DI) ± 1 or 2	CF, PF, AF, ZF, SF, OF
SCAS	Scan string	SCASB/SCASW	Set flags as per (AL or AX) − ((ES)0 + (DI)) (DI) ← (DI) ± 1 or 2	CF, PF, AF, ZF, SF, OF
LODS	Load string	LODSB/LODSW	(AL or AX) ← ((DS)0 + (SI)) (SI) ← (SI) ± 1 or 2	None
STOS	Store string	STOSB/STOSW	((ES)0 + (DI)) ← (AL or AX) ± 1 or 2 (DI) ← (DI) ± 1 or 2	None

Prefix	Used with:	Meaning
REP	MOVS STOS	Repeat while not end of string CX ≠ 0
REPE/REPZ	CMPS SCAS	Repeat while not end of string and strings are equal CX ≠ 0 and ZF = 1
REPNE/REPNZ	CMPS SCAS	Repeat while not end of string and strings are not equal CX ≠ 0 and ZF = 0

Mnemonic	Meaning	Format	Operation	Flags Affected
CLD	Clear DF	CLD	(DF) ← 0	DF
STD	Set DF	STD	(DF) ← 1	DF

Continued ...

Appendix B

MMX Instruction Set Summary

Mnemonic	Name
MOVQ	Move quad word
MOVD	Move double word
PADDB	Packed add bytes
PADDW	Packed add words
PADDD	Packed add double words
PADDSB	Packed add signed bytes
PADDSW	Packed add signed words
PADDUSB	Packed add unsigned bytes
PADDUSW	Packed add unsigned words
PSUBB	Packed subtract bytes
PSUBW	Packed subtract words
PSUBD	Packed subtract double words
PSUBSB	Packed subtract signed bytes
PSUBSW	Packed subtract signed words
PSUBSD	Packed subtract signed double words
PSUBUSB	Packed subtract unsigned bytes
PSUBUSW	Packed subtract unsigned words
PMULLW	Packed multiply low words
PMULHW	Packed multiply high words
PMADDWD	Packed multiply and add words
PSRAW	Packed shift right arithmetic words
PSRAD	Packed shift right arithmetic double words
PCMPEQB	Packed compare for equal bytes
PCMPEQW	Packed compare for equal words
PCMPEQD	Packed compare for equal double words
PCMPGTB	Packed compare for greater than bytes
PCMPGTW	Packed compare for greater than words
PCMPGTD	Packed compare for greater than double words
PAND	Bitwise logical AND
PANDN	Bitwise logical AND NOT
POR	Bitwise logical OR
PXOR	Bitwise logical exclusive-OR
PSLLW	Packed shift left logical word
PSLLD	Packed shift left logical double word
PSLLQ	Packed shift left logical quad word
PSRLW	Packed shift right logical word
PSRLD	Packed shift right logical double word
PSRLQ	Packed shift right logical quad word
PACKUSWB	Pack words to bytes with unsigned saturation
PACKSSWB	Pack words to bytes with signed saturation
PACKSSDW	Pack double words to words with signed saturation
PUNPCKHBW	Unpack high packed data bytes to words
PUNPCKHWD	Unpack high packed data words to double words
PUNPCKHDQ	Unpack high packed data double words to quad word
PUNPCKLBW	Unpack low packed data bytes to words
PUNPCKLWD	Unpack low packed data words to double words
PUNPCKLDQ	Unpack low packed data double words to quad word

Continued ...

Appendix C

Summary of Selected FPU Instructions

Mnemonic	Description
FINIT	Initialize floating-point unit
FLD	Load floating-point value onto FPU register stack
FSTP	Store floating-point value and pop the FPU register stack
FADD	Add floating-point values
FSUB	Subtract floating-point values
FMUL	Multiply floating-point values
FDIV	Divide floating-point values

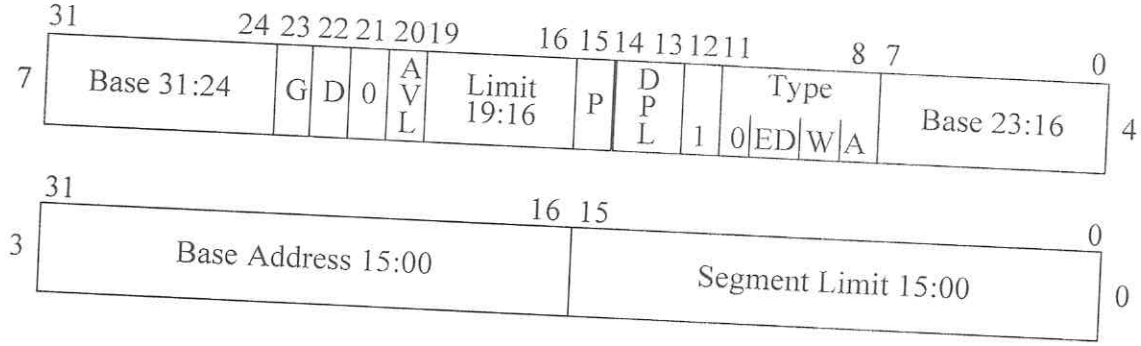
Summary of Selected SSE Instructions

Mnemonic	Description
MOVAPS	Move aligned packed single-precision floating-point values
MOVUPS	Move unaligned packed single-precision, floating-point
MOVHPS	Move high packed single-precision floating-point
MOVHLPS	Move packed single-precision floating-point high to low
MOVLHPS	Move packed single-precision floating-point low to high
MOVLPS	Move low packed single-precision floating-point
MOVMSKPS	Move packed single-precision floating-point mask
MOVSS	Move scalar single-precision floating-point
ADDSS	Scalar add pair of operands
ADDPS	Packed add pairs of operands
SUBSS	Scalar subtract pair of operands
SUBPS	Packed subtract pairs of operands
MULSS	Scalar multiply pair of operands
MULPS	Packed multiply pairs of operands
DIVSS	Scalar divide pair of operands
DIVPS	Packed divide pairs of operands

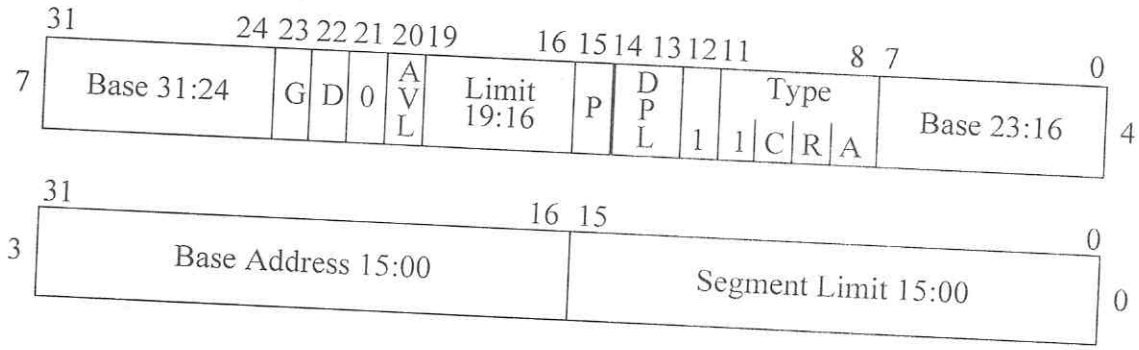
Continued ...

Appendix D

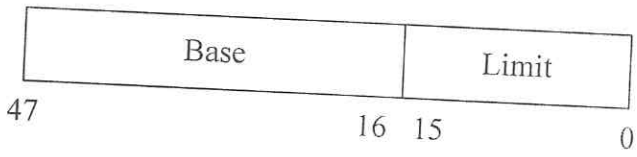
Data-Segment Descriptor:



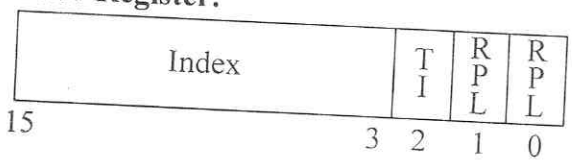
Code-Segment Descriptor:



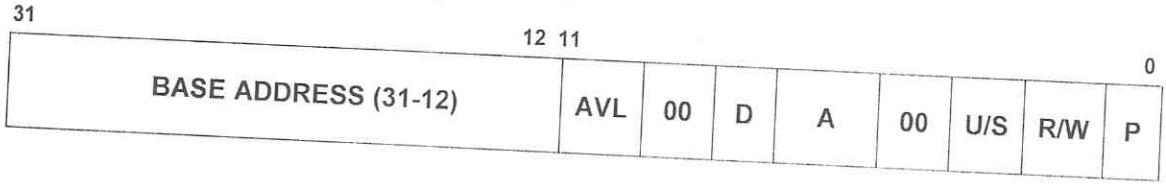
Global Descriptor Table Register (GDTR) and Interrupt Descriptor Table Register (IDTR):



Selector Register:



Directory or Page Table Entry Format:



Continued ...

Appendix E
ASCII Chart:

DEC	OCT	HEX	Symbol	DEC	OCT	HEX	Symbol	DEC	OCT	HEX	Symbol	DEC	OCT	HEX	Symbol
0	0	0	NUL	32	40	20	SPACE	64	100	40	@	96	140	60	`
1	1	1	SOH	33	41	21	!	65	101	41	A	97	141	61	a
2	2	2	STX	34	42	22	"	66	102	42	B	98	142	62	b
3	3	3	ETX	35	43	23	#	67	103	43	C	99	143	63	c
4	4	4	EOT	36	44	24	$	68	104	44	D	100	144	64	d
5	5	5	ENQ	37	45	25	%	69	105	45	E	101	145	65	e
6	6	6	ACK	38	46	26	&	70	106	46	F	102	146	66	f
7	7	7	BEL	39	47	27	'	71	107	47	G	103	147	67	g
8	10	8	BS	40	50	28	(	72	110	48	H	104	150	68	h
9	11	9	HT	41	51	29	)	73	111	49	I	105	151	69	i
10	12	0A	LF	42	52	2A	*	74	112	4A	J	106	152	6A	j
11	13	0B	VT	43	53	2B	+	75	113	4B	K	107	153	6B	k
12	14	0C	FF	44	54	2C	,	76	114	4C	L	108	154	6C	l
13	15	0D	CR	45	55	2D	-	77	115	4D	M	109	155	6D	m
14	16	0E	SO	46	56	2E	.	78	116	4E	N	110	156	6E	n
15	17	0F	SI	47	57	2F	/	79	117	4F	O	111	157	6F	o
16	20	10	DLE	48	60	30	0	80	120	50	P	112	160	70	p
17	21	11	DC1	49	61	31	1	81	121	51	Q	113	161	71	q
18	22	12	DC2	50	62	32	2	82	122	52	R	114	162	72	r
19	23	13	DC3	51	63	33	3	83	123	53	S	115	163	73	s
20	24	14	DC4	52	64	34	4	84	124	54	T	116	164	74	t
21	25	15	NAK	53	65	35	5	85	125	55	U	117	165	75	u
22	26	16	SYN	54	66	36	6	86	126	56	V	118	166	76	v
23	27	17	ETB	55	67	37	7	87	127	57	W	119	167	77	w
24	30	18	CAN	56	70	38	8	88	130	58	X	120	170	78	x
25	31	19	EM	57	71	39	9	89	131	59	Y	121	171	79	y
26	32	1A	SUB	58	72	3A	:	90	132	5A	Z	122	172	7A	z
27	33	1B	ESC	59	73	3B	;	91	133	5B	[	123	173	7B	{
28	34	1C	FS	60	74	3C	<	92	134	5C	\	124	174	7C	|
29	35	1D	GS	61	75	3D	=	93	135	5D	]	125	175	7D	}
30	36	1E	RS	62	76	3E	>	94	136	5E	^	126	176	7E	~
31	37	1F	US	63	77	3F	?	95	137	5F	_	127	177	7F	

DEC	OCT	HEX	Symbol	DEC	OCT	HEX	Symbol	DEC	OCT	HEX	Symbol	DEC	OCT	HEX	Symbol
128	200	80	€	160	240	A0		192	300	C0	À	224	340	E0	à
129	201	81		161	241	A1	¡	193	301	C1	Á	225	341	E1	á
130	202	82	‚	162	242	A2	¢	194	302	C2	Â	226	342	E2	â
131	203	83	ƒ	163	243	A3	£	195	303	C3	Ã	227	343	E3	ã
132	204	84	„	164	244	A4	¤	196	304	C4	Ä	228	344	E4	ä
133	205	85	…	165	245	A5	¥	197	305	C5	Å	229	345	E5	å
134	206	86	†	166	246	A6	¦	198	306	C6	Æ	230	346	E6	æ
135	207	87	‡	167	247	A7	§	199	307	C7	Ç	231	347	E7	ç
136	210	88	ˆ	168	250	A8	¨	200	310	C8	È	232	350	E8	è
137	211	89	‰	169	251	A9	©	201	311	C9	É	233	351	E9	é
138	212	8A	Š	170	252	AA	ª	202	312	CA	Ê	234	352	EA	ê
139	213	8B	‹	171	253	AB	«	203	313	CB	Ë	235	353	EB	ë
140	214	8C	Œ	172	254	AC	¬	204	314	CC	Ì	236	354	EC	ì
141	215	8D		173	255	AD		205	315	CD	Í	237	355	ED	í
142	216	8E	Ž	174	256	AE	®	206	316	CE	Î	238	356	EE	î
143	217	8F		175	257	AF	¯	207	317	CF	Ï	239	357	EF	ï
144	220	90		176	260	B0	°	208	320	D0	Ð	240	360	F0	ð
145	221	91	‘	177	261	B1	±	209	321	D1	Ñ	241	361	F1	ñ
146	222	92	’	178	262	B2	²	210	322	D2	Ò	242	362	F2	ò
147	223	93	“	179	263	B3	³	211	323	D3	Ó	243	363	F3	ó
148	224	94	”	180	264	B4	´	212	324	D4	Ô	244	364	F4	ô
149	225	95	•	181	265	B5	µ	213	325	D5	Õ	245	365	F5	õ
150	226	96	–	182	266	B6	¶	214	326	D6	Ö	246	366	F6	ö
151	227	97	—	183	267	B7	·	215	327	D7	×	247	367	F7	÷
152	230	98	˜	184	270	B8	¸	216	330	D8	Ø	248	370	F8	ø
153	231	99	™	185	271	B9	¹	217	331	D9	Ù	249	371	F9	ù
154	232	9A	š	186	272	BA	º	218	332	DA	Ú	250	372	FA	ú
155	233	9B	›	187	273	BB	»	219	333	DB	Û	251	373	FB	û
156	234	9C	œ	188	274	BC	¼	220	334	DC	Ü	252	374	FC	ü
157	235	9D		189	275	BD	½	221	335	DD	Ý	253	375	FD	ý
158	236	9E	ž	190	276	BE	¾	222	336	DE	Þ	254	376	FE	þ
159	237	9F	Ÿ	191	277	BF	¿	223	337	DF	ß	255	377	FF	ÿ

End of Paper